

EXHIBIT J

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:)	Atty. Docket No.:
ROCHETTE ET AL.)	78802 (120-1 US)
)	
Serial No. 10/939,903)	Art Unit: 2139
)	
Filing Date: SEPTEMBER 13, 2004)	Examiner:
)	RONALD BAUM
Confirmation No. 5216)	
)	
For: SYSTEM FOR CONTAINERIZATION)	
OF APPLICATION SETS)	
)	

AMENDMENT

EFILED

Commissioner of Patents

Dear Sir:

In response to the Office Action dated June 3, 2008
please amend the above-identified application as follows:

Amendments to the Claims are reflected in the listing of
claims, which begins on page 2 of this paper.

Remarks begin on page 10 of this paper.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

Amendments to the Claims

1. (currently amended) In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications ~~[[may be]]~~are executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, the method ~~comprising the steps of:~~

storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel, ~~and~~ wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server prior to said storing step, wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server, and wherein the application software cannot be shared

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

between the plurality of secure containers of application software, and wherein each of the containers has a unique root file system that is different from an operating system's root file system.

2. (original) A method as defined in claim 1, wherein each container has an execution file associated therewith for starting the one or more applications.
3. (original) A method as defined in claim 2, wherein the execution file includes instructions related to an order in which executable applications within will be executed.
4. (original) A method as defined in claim 1 further comprising the step of pre-identifying applications and system files required for association with the one or more containers prior to said storing step.
5. (currently amended) A method as defined in claim 2, further comprising the step of modifying at least some of the associated system files in plural containers to provide an association with a container specific identity assigned to ~~the~~ a particular container.
6. (original) A method as defined in claim 2, comprising the step of assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address.
7. (original) A method as defined in claim 2 further comprising the step of modifying at least some of the system

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

files to define container specific mount points associated with the container.

8. (original) A method as defined in claim 1, wherein the one or more applications and associated system files are retrieved from a computer system having a plurality of secure containers.

9. (original) A method as defined in claim 2, wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation is associated with at least some of the containers prior to the applications within the containers being executed.

10. (currently amended) A method as defined in claim 2, wherein in operation when ~~applications~~ an application residing within a container ~~are~~ is executed, ~~applications~~ said application ~~within a container have~~ has no access to system files or applications in other containers or to system files within the operating system during execution thereof ~~if those applications or system files are read/write files.~~

11. (original) A method as defined in claim 2, wherein containers include files stored in network file storage, and parameters forming descriptors of containers stored in a separate location.

12. (original) A method as defined in claim 11, further comprising the step of merging the files stored in network storage with the parameters to affect the step of storing in claim 1.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

13. (currently amended) A method as defined in claim 1 further comprising the step of associating with a ~~container~~ plurality of containers a stored history of when processes related to applications within the container are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application.

14. (currently amended) A method as defined in claim 1 comprising the step of creating containers prior to said step of storing containers in memory, wherein containers are created by:

- a) running an instance of a service on a server;
- b) determining which files are being used; and,
- c) copying applications and associated system files to memory without overwriting the associated system files so as to provide a second instance of the applications and associated system files.

15. (original) A method as defined in claim 14 comprising the steps of:
assigning an identity to the containers including at least one of a unique IP address, a unique Mac address and an estimated resource allocation;
installing the container on a server; and,
testing the applications and files within the container.

16. (original) A method as defined in claim 1 comprising the step of creating containers prior to said step of storing containers in memory, wherein a step of creating containers includes:
using a skeleton set of system files as a container starting point and installing applications into that set of files.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

17. (currently amended) A computing system for performing a plurality of tasks each comprising a plurality of processes comprising:

a system having a plurality of secure ~~stored~~ containers of associated files accessible to, and for execution on, one or more servers, each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers, each container of files ~~having~~ is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac_address; wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes wherein the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system, each container having its own execution file associated therewith for starting one or more applications, in operation, each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and, a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.

18. (original) A computing system as defined in claim 17, further comprising a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

19. (original) A computing system as defined in claim 18, wherein the run time module includes an intercepting module associated with the plurality of containers for intercepting system calls from any of the plurality of containers and for providing values alternate to values the kernel would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of the IP address, the host name and the Mac_Address.

20. (original) A computing system as defined in claim 17, wherein the run time module performs:
monitoring resource usage of applications executing;
intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;
comparing the monitored resource usage of the at least one respective application with the resource limits; and,
forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

21. (currently amended) A method as defined in claim 1 further comprising ~~the step of~~ installing a service on a target server selected from one of the plurality of servers, wherein ~~the step of~~ installing the service includes ~~the steps of~~:
using a graphical user interface, associating a unique icon representing a service with an unique icon representing a server for hosting applications related to the service and for executing the service, so as to cause the applications to be distributed to, and installed on the target server.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

22. (original) A method as defined in claim 21 wherein the target server and the graphical user interface are at remote locations.

23. (original) A method as defined in claim 22, wherein the graphical user interface is installed on a computing platform, and wherein the computing platform is a different computing platform than the target server.

24. (original) A method as defined in claim 23, wherein the step of associating includes the step of relatively moving the unique icon representing the service to the unique icon representing a server.

25. (original) A method claim 21, further comprising, the step of testing to determine if the selected target server is a valid computing platform, prior to causing the applications to be distributed to, and installed on the target server.

26. (currently amended) A method according to claim 21 further comprising, ~~the step of~~ creating a user account for the service.

27. (original) A method as defined in claim 21, further comprising the step of installing files specific to the selected application on the selected server.

28. (original) A method according to claim 21 further comprising the steps of setting file access permissions to allow a user to access the one of the applications to be distributed.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

29. (currently amended) A method as defined in claim 24 further comprising ~~the step of~~ starting a distributed software application.

30. (currently amended) A method according to anyone of claims 24 further comprising ~~the steps of~~ updating a console on the selected target server with information indicating that the service is resident on the selected target server.

31. (currently amended) A method as defined in claim 1, further comprising ~~the step of:~~ de-installing a service from a server, comprising ~~the steps of:~~
displaying the icon representing the service;
displaying [[a]]the icon representing the server on which the service is installed;
and utilizing the icon representing the service and the icon representing the server to initiating the de-installation of the selected service from the server on which it was installed.

32. (currently amended) A method according to claim 31 further comprising ~~the step of~~ separating icon representing the service from the icon representing the server.

33. (currently amended) A method according to claim 31 further comprising ~~the step of~~ testing whether the selected server is a valid computing platform for de-installation of the service.

34. (currently amended) A method according to claim 31 further comprising ~~the step of~~ copying data file changes specific to the service back to a storage medium from which the data file changes originated prior to installation.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

REMARKS

Claims 1-34 are pending in this application and are rejected under 35 U.S.C. 102 (b) as being anticipated by U.S. Patent 6,381,742 B1 ("Forbes et al.").

For a claim to be anticipated, all of the limitations of the claim must be present in a single reference.

Claim 1 has been amended to include recitations which more clearly define the invention and which also distinguish Applicants' invention from the teaching of Forbes et al.

Amended Claim 1 now recites:

1. In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel **a set of associated local system files compatible** with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications may be executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising the steps of:

storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

execute the one or more applications, for use with a local kernel residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel, ~~and~~ wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server prior to said storing step., wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server, and wherein the application software cannot be shared between the plurality of secure containers of application software, and wherein each of the containers has a unique root file system that is different from an operating system's root file system.

Support for the amendments in claim 1 can be found within the specification.

For example:

At paragraph [54] the following text is found:

"In embodiments of the invention, all applications within a given container have their own common root file system exclusive thereto and unique from other containers and from the operating system's root file system."

Figure 6 illustrates that system files from the underlying operating system are used independent of system files in the container. In the instant invention system files

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

within the container are application specific and are distinct from and system files within the OS.

Figure 9 shows that system files from the underlying operating system are used by operating system applications wherein system files from the container are used by applications associated with the container.

At paragraph [91] the following text is recited:

"Turning now to Figure 1, a system is shown having a plurality of servers 10a, 10b. Each server includes a processor and an independent operating system. Each operating system includes a kernel 12, hardware 13 in the form of a processor and a set of associated system files, not shown. Each server with an operating system installed is capable of executing a number of application programs. Unlike typical systems, containerized applications are shown on each server having application programs 11a, 11b and related system files 11c. These system files are used in place of system files such as library and configuration files present typically used in conjunction with the execution of applications."

(underlining added)

At paragraph [92], the following text is recited:

"Figure 2 illustrates a system in accordance with the invention in which multiple applications 21a, 21b, 21c, 21d, 21e, and 21f can execute in a secure environment on a single server. This is made possible by associating applications with secure containers 20a, 20b and 20c.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

Applications are segregated and execute with their own copies of files and with a unique network identity. In this manner multiple applications may contend for common resources and utilize different versions of system files. Moreover, applications executing within a secure container can co-exist with applications that are not associated with a container, but which are associated with the underlying operating system." (underlining added)

At paragraph [94]the following text is recited:

"The Secure application containers 20a, 20b and 20c are shown in Figure 2 in which each combines un-installed application files on a storage medium or network, application files installed on a computer, and system files. The container is an aggregate of all files required to successfully execute a set of software applications on a computing platform. In embodiments of the invention, containers are created by combining application files with system files."

Paragraph [107] clearly teaches the relationship between files and containers:

"It can be seen from Figure 6 that each application executing associated with a container 20a or 20b, or contained within a container 20a or 20b, is able to accesses files that are dedicated to the container. Applications with a container association, that is, applications contained within a container, are not able to access the files provided with the underlying

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

operating system of the compute platform 40 upon which they execute. Moreover, applications with a container 20a association are not able to access the files contained in another container 20b." (underlining added)

At paragraphs 112, 117, 121, 122, and 131, the following text is found which supports the amendments to Claim 1.

[112] In a first embodiment all files are exclusive. That is, each container has a separate physical copy of the files required by applications associated with the container. In future embodiments any files that is read-only will be shared between containers.
(underlining added)

[117] Figure 3 illustrates container creation. The result of container creation is the collection of files required by applications associated with the container and the assembly of container specific configuration information. Container files include both those files provided by an operating system distribution and those files provided by an application install media or package file.
(underlining added)

[121] Referring once again to Figure 3, applications may come from third party installation media on CDROM, package files 30, or as downloads from a web site, etc. Once installed in the container the applications become unique to the container in the manner described way that has been described heretofore.
(underlining added)

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

[122] The second method employed to create a container file system uses an existing server, for example a computer or server already installed at a customer site. This is also shown in Figure 3. The applications of interest may have been installed on an existing server before the introduction of the software and data structures in accordance with this invention. Files are copied from the existing server 32, including system files and application specific files to network storage. Once the files are copied from the existing server a container is created from those files.

(underlining added)

[131] In some embodiments of the invention, the method involves installing at least one of the pluralities of applications in a container's own root file system.

(underlining added)

Applicants would respectfully like to point out one significant distinction between the invention of Forbes et al. and the instant invention defined in the amended claims of this invention; and more particularly amended Claim 1.

Applicants' claimed invention is defined as having a plurality of servers with operating systems that differ, operating in disparate computing environments. Applicants' claimed invention is defined as having "secure" containers of application software which include OS files that are copies or modified copies of OS files while preserving the original OS files by "not" overwriting them. These OS file copies reside with each container and are modified if necessary so as to be compatible with the applications that are to be run from any

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

given container. That is to say one container may be running an application that is compatible with Solaris 8, and another container may be running an application that has been written for and compatible with Solaris 10. Therefore, different containers on a single server are capable of running applications designed for different operating systems.

In contrast, Forbes et al. provide a software package manager that uses a distribution unit containing components for a software package and a manifest file that describes the distribution unit to manage the installation, execution, and uninstallation of software packages on a computer. Information in the manifest file pertaining to a software package is stored in a code store data structure upon installation of the package. The manifest file also contains information that permits the software package manager to resolve any software dependencies upon installation. The software package manager uses the code store data structure to locate the required components when the software is executed and to remove the components appropriately when the software is uninstalled.

Therefore, Forbes et al. teach a software package manager for running a software package on a system for which it was intended to run. More particularly, if the system was written or coded to be run on Windows XP, Forbes et al.'s package manager can execute on Windows XP.

Since the teaching of Forbes et al. directly instructs replacing older versions of software with newer versions by writing over older versions with newer versions, Forbes et al.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

are not capable of having different versions run under the same Operating System together.

Applicants provide containers that are isolated and wherein each container has the necessary operating system files for an application to run and wherein different containers may have different versions of the same software.

Forbes et al. do not allow this. Forbes et al. teach replacement of older files with newer versions.

There is not a hint or a suggestion of having different versions of the same software on the system at the same time. There is no hint or suggestion of the provision of isolated containers of software to allow this.

Turning now to Fig. 3b of Forbes et al., the drawing (below) illustrates that contrary to Applicants' claimed invention, files are not copies in the Forbes et al. patent, they are replacements. This difference is important. Replacing an operating system file as Forbes et al. do, would not allow Applicants' claimed invention to function in the way in which it was designed.

In Forbes et al., boxes 343 and 353 show removing an older file and replacing that older file with an updated file. Boxes 359 and 361 indicate the same; "remove and replace". Since these files are removed and replaced, Forbes et al. do not use an isolated "copy" that is stored within a container. More particularly, Forbes et al. do not teach multiple

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

containers each having isolated copies or modified copies of system files in each container.

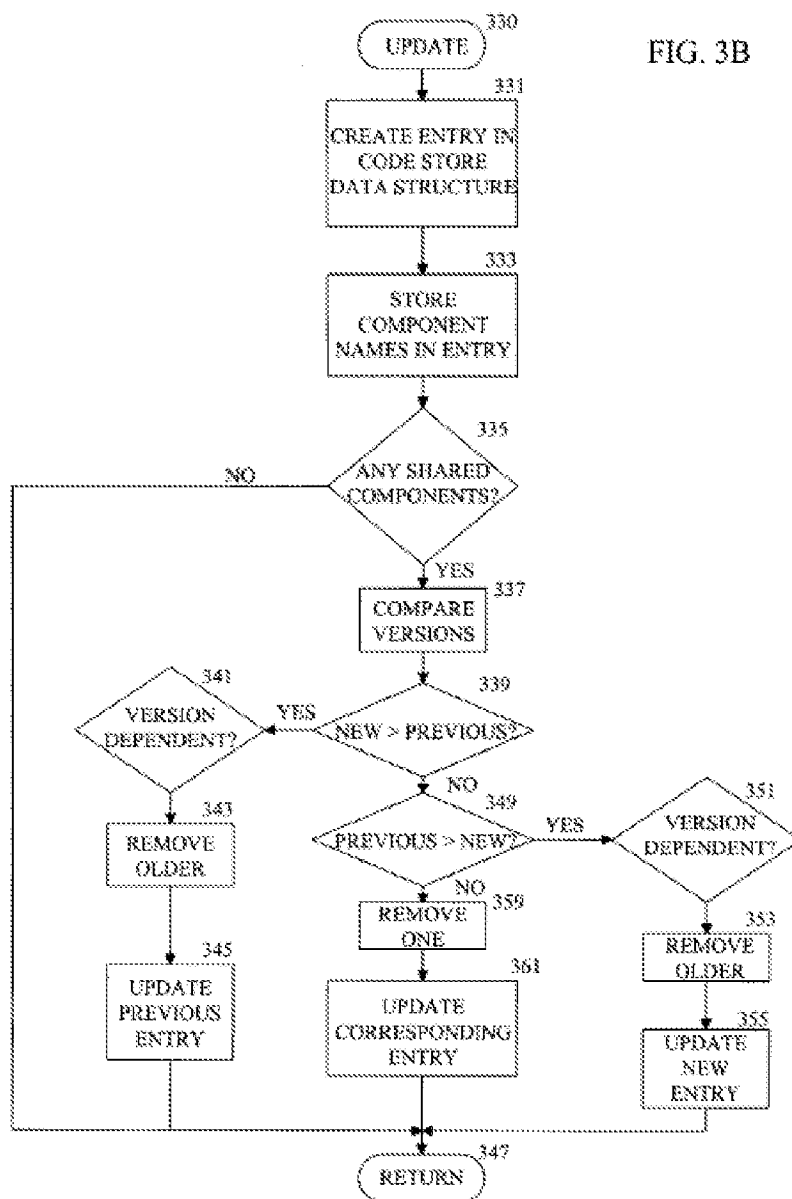
Not only do Forbes et al. not have the required isolation afforded by the method defined in amended Claim 1, by writing over or removing a file, the original files are not available for use in building/creating other containers. Since containers cannot share application files in Applicants' claimed invention, Forbes et al. teach a system which would not afford a plurality of isolated different containers to be created. For containers to be isolated from one another, application files cannot be shared between them. Forbes et al have a single instance installation of a single software program and are not concerned with multiple software programs in different containers.

In re Patent Application of:

ROCHETTE ET AL.Serial No. **10/939,903**Filed: **09/13/2004****U.S. Patent**

Apr. 30, 2002

Sheet 6 of 8

US 6,381,742 B2**FIG. 3B**

Further instances that files are not copied but are replaced in Forbes et al.'s system are found at column 2, line 66, through column 3, line 3.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

"During the installation, the package manager can optionally scan the code store data structure to determine if a component to be installed already exists on the computer and updates the code store data structure with the location of the later version of the component."

Col 6 line 65 through col 7 line 2 states "Fred's Software Company also creates a "manifest" file 207 describing the CoolestApp. The CoolestApp manifest file 207 contains information about CoolestApp, including the name of the CoolestApp distribution unit 209, the version number of the software package (all components in the distribution unit 209 have the same version number in this embodiment), and the operating systems under which the CoolestApp executes. Fred's Software Company bundles the CoolestApp distribution unit 209 and manifest file 207 into a distribution unit file 205 for storage on the server 201."

The paragraph above from Forbes et al. indicates that the Operating System must be compatible with the software application. It specifies that the software application is tied to the version of the Operating System.

Thus, in Forbes et al., files being installed must be consistent with the underlying OS.

In Forbes et al., the files, and hence the application being installed, are used with the existing OS. The files are not stored into an isolated container as there are not multiple containers of application software. The files in

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

Forbes et al. are not isolated from the OS; and they are not able to run on an incompatible OS or version of OS.

Col 9 lines 13-18 states: "The package manager checks the name and version of the software package contained in the manifest file against the code store data structure to determine if the software package has already been installed (step 303). If so, the package manager exits (step 321)."

Thus, Forbes et al. teach that if the files exist on the system then the install exits. That is, one instance of the application exists. The files do not exist in isolated containers nor do multiple versions/copies exist nor are multiples capable of executing correctly.

Col 10, lines 20-43, states: "If the newly stored component is a newer version of the previously stored component (step 339) and the code store data structure entry for the previously installed software package that references the older version does not indicate it is version dependent (step 341), the package manager removes the older version from the directory in which it is stored (step 343) and updates the code store data structure entry for the previously installed software package to point to the newer version (step 345). If there is a version dependency noted, the package manager leaves the older version in the directory (step 347).

If the newly stored component is older than the previously stored component (step 349) and the software package does not indicate a version dependency (step 351), the package manager removes the older version from the newly created directory (step 353) and updates the code store data

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

structure entry for the newly installed software package to point to the newer version (step 355). As before, if there is a version dependency noted, the package manager does nothing to the older version (step 347).

If the components are the same version, the package manager chooses one to remove from its directory (step 359) and updates the corresponding entry to point to the other component (step 361).” (underlining added).

This text from Forbes et al., clearly teaches that newer versions overwrite existing older versions and so forth. In Forbes et al., a single “file” is the result. Forbes et al. do not teach that multiple copies of system files are used. There is no definition of an isolated environment in which applications execute.

In contrast, the instant invention, as defined in amended Claim 1, recites that containers have a unique root file system such that container files are distinct from other applications and from the operating system software.

Therefore, the instant invention as defined in Claim 1, creates the ability for multiple applications, including applications of the same type, to be able to execute correctly on the same Operating System. By way of example, multiple Oracle database managers are able to execute correctly on the same operating system in a container context as defined in Claim 1. This is not possible in the Forbes et al. invention.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

It is significant that multiple Oracle/SQL Server/Sybase/CRM/App servers/ etc. can execute correctly on the same Operating System. The number of OS images needed, along with the associated license and management costs are greatly diminished. This enables a new and unique ability for combining, deploying and managing applications.

Claim 5 defines:

A method as defined in claim 2, further comprising the step of modifying at least some of the associated system files in plural containers to provide an association with a container specific identity assigned to ~~the~~ a particular container.

With regard to amended claim 5, Forbes et al. do not have plural containers of application software as defined in Claims 1 or 2 and do not provide an association with a container specific identity. Forbes et al. only teach having a single application not plural applications and make no mention of a container specific identity.

Claim 6 defines:

A method as defined in claim 2, comprising the step of assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address.

The Examiner states in the Office Action that

"The teachings of Forbes et al (Forbes et al Abstract, col 2 lines 33-col. 3 line 56, figures 1-4 and associated

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

descriptions, whereas the use of a software package manager in a network environment using a distribution unit containing components for a software package and manifest file that supports installation, execution/runtime aspects (i.e. assigning a unique identity..." of predetermined applications/associated services across all types of code/operating systems, so as to allow easily extending new code types as they arise (e.g., Forbes et al col. 2 lines 38-col 3 lines 49 clearly encompasses the limitations as broadly interpreted by the examiner) suggest such limitations."

Applicants would respectfully like to point out that Claim 6 importing the limitations of amended claim 1 recites:

In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel **a set of associated local system files compatible** with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications may be executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising the steps of:

storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications,

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

for use with a local kernel residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel, wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server., wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server, and wherein files cannot be shared between the plurality of secure containers of application software, and wherein each of the containers each has a unique root file system that is different from the operating system's root file system, wherein each container has an execution file associated therewith for starting the one or more applications and further comprising the step of assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address.

Many of the differences in Claim 6 and the teachings of Forbes et al. have been discussed above in this response, with regard to Claim 1.

However, Forbes et al. do not teach:

- Providing secure containers;
- Containers that have copies of OS files

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

- Containers having a unique identity and having a unique root file system different from the root file system of the underlying OS under which the containers run; and
- assigning a unique associated identity to each of a plurality of containers, since Forbes doesn't teach "a plurality of containers" wherein each identity includes an IP address, host name or Mac Address.

The Examiner repeatedly points to col. 2 and col. 3 of Forbes et al. for teaching of the present invention. However, Applicants cannot find such teaching, and if the Examiner is of the opinion that this claim is anticipated, Applicants would be appreciative if the Examiner would indicate "specifically" where these limitations are found in Forbes et al.

In contrast, Forbes et al. teach the installation of a single software program into a system for which the software and OS are compatible. There is no hint or suggestion in Forbes et al. of storing in containers modified copies of OS system files so that software not designed for a particular OS can run.

Claim 7 defines:

A method as defined in claim 2 further comprising the step of modifying at least some of the system files to define container specific mount points associated with the container.

The Examiner has used essentially the same passages (below) from col. 2 to col. 3 to suggest that the Claim is anticipated.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

The teachings of Forbes et al (Forbes et al Abstract, col. 2, lines 33-col. 3, line 56, figures 1-4 and associated descriptions, whereas the use of a software package manager in a network environment using a distribution unit containing components for a software package and manifest file that supports the installation, execution/runtime aspects (i.e., 'modifying at least some of the system files ... define container specific mount points') of predetermined applications /associated services across all types of code/operating systems, so as to allow easily extending new code types as they arise (e.g., Forbes et al col. 2, lines 38-col. 3, lines 49), clearly encompasses the claimed limitations as broadly interpreted by the examiner.) suggest such limitations.

Notwithstanding, Forbes et al. do not provide the steps of:

"storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel, wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server., wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server, and wherein files cannot be shared between the plurality of secure containers of

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

application software, and wherein each of the containers each has a unique root file system that is different from the operating system's root file system, wherein each container has an execution file associated therewith for starting the one or more applications."

Furthermore and more importantly, with reference to Claim 7, Forbes et al. do not modify at least some of the system files to define container specific mount points associated with the container. As was pointed out earlier Forbes et al. do not provide containers.

Claim 9 is "a method as defined in claim 2, wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation is associated with at least some of the containers prior to the applications within the containers being executed."

Claim 9 imports all of the limitations of amended Claim 1 and claim 2 from which it depends. Furthermore, Claim 9 defines within the method of Claim 2, server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation is associated with at least some of the containers prior to the applications within the containers being executed.

There is no such disclosure in Forbes et al. and Applicants would appreciate if the Examiner could point the Applicants to such disclosure by indicating the specific page and line number where this is taught. A careful read of col.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

2 and col. 3 and the figures does not disclose the claimed limitations.

The Examiner has provided essentially the same general statement below:

The teachings of Forbes et al (Forbes et al Abstract, col. 2, lines 33-col. 3, line 56, figures 1-4 and associated descriptions, whereas the use of a software package manager in a network environment using a distribution unit containing components for a software package and manifest file that supports the installation, execution/runtime aspects (i.e., 'server information related to hardware resource usage ... associated with ... prior to the applications ... executed') of predetermined applications /associated services across all types of code/operating systems, so as to allow easily extending new code types as they arise (e.g., Forbes et al col. 2, lines 38-col. 3, lines 49), clearly encompasses the claimed limitations as broadly interpreted by the examiner.) suggest such limitations.

However, the teachings of Forbes et al. in the Abstract and in Col. 2, line 33 to Col. 3, line 56, only disclose the following:

Abstract

A software package manager uses a distribution unit containing components for a software package and a manifest file that describes the distribution unit to manage the installation, execution, and uninstallation of software packages on a computer. Information in the manifest file pertaining to a software package is stored in a code store data structure upon installation of the package. The manifest file also contains information that permits the software package manager to resolve any software dependencies upon installation. The software package manager uses the code store data structure to locate the required components when the software is executed and to remove the components appropriately when the software is uninstalled.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

Col. 2, line 33 to Col. 3, line 56

The above-mentioned shortcomings, disadvantages and problems are addressed by the present invention, which will be understood by reading and studying the following specification.

A software package manager uses a distribution unit containing components for a software package and a manifest file that describes the distribution unit to manage the installation, execution, and uninstallation of software packages on a computer. For installation, the package manager acquires the manifest file and parses it to learn if the software package depends on any additional components. The package manager resolves any dependencies by acquiring a distribution unit containing the needed component and installs the dependency's distribution unit as described below. Because dependencies can be nested within dependencies, the package manager recursively processes all the dependencies before finishing the installation of the software package that depends upon the additional components.

The software package manager acquires the distribution unit and extracts the components in the distribution unit into a directory on the computer. The package manager causes the operating system of the computer to install the software. The package manager then updates a code store data structure with information in the manifest file. The fields in the code store data structure contains such information as the name and version of the distribution unit, a list of the components and their location on the computer, and the source of the distribution unit. Additional fields in the code store data structure can also contain a component version, a component data type, and a digital signature if one was affixed to the distribution unit.

During the installation, the package manager can optionally scan the code store data structure to determine if a component to be installed already exists on the computer and updates the code store data structure with the location of the later version of the component.

When a user requests execution of software, the package manager uses the code store data structure to locate the appropriate components for the operating system to use. When the user requests the uninstallation of a software package, the package manager deletes the appropriate components from

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

the computer and updates the code store data structure accordingly.

The manifest file and distribution unit optionally are combined into a distribution unit file.

The manifest file format is common across all types of code and operating systems and easily extended to embrace new code types as they arise. The manifest file and distribution unit can be stored on all types of media from traditional magnetic and optical disks to networked servers. The distribution units for dependencies do not have to reside on the same type of media as the distribution unit or the manifest file that refers to the dependency. More than one distribution unit can be resident in a distribution unit file and a distribution unit file can contain a mixture of distribution units containing different code types.

Thus, the software package manager, the manifest file, the distribution unit and the code store data structure of the present invention solve the problems with existing distribution mechanisms. The manifest file is not particular to a particular code type or operating system and allows for the specification of nested software dependencies. Because the manifest file contains the location of the distribution units for any dependencies, the software package manager can acquire and install the dependencies without requiring manual intervention by the user. Different types of distribution units can be mixed in a distribution unit file so that a single mechanism is used to acquire and install all types of code.

The code store data structure maintained by the software package manager contains information about the installed software such as version and installation location, and is used to resolve version discrepancies among software programs that share components. The code store data structure is used by the package manager to locate necessary components when the software is executed so that a component stored in one directory can be readily shared by software programs with components in different directories. Finally, the code store data structure eases the uninstallation process by centralizing all the information about installed components.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

Once again, it must be emphasized that there is no method of containerizing software applications in containers wherein the limitations of amended Claims 1, 2 and 9 are met. The Examiner recites the text above and states that the limitations of Claim 9 are present, however they are not.

Applicants believe that the amendments to Claim 1 clearly distinguish Claim 9 from the cited Forbes et al. reference.

If the Examiner is of the opinion that amended Claim 1 is anticipated by Forbes et al., Applicants would appreciate if the Examiner would specifically point out where each limitation in amended base Claim 1 is found in the text above. Furthermore, Applicants would be appreciative if the Examiner would point out where the limitations of Claim 2 and 9 can be found. It is not sufficient or instructive to Applicants to recite that the text above clearly encompasses the claimed limitations, as broadly interpreted by the Examiner. It is not clear, nor is it evident, that the claimed features are disclosed by Forbes et al.

Claim 10 has been amended to more clearly define the invention.

10. (amended) A method as defined in claim 2, wherein in operation when an application residing within a container ~~are~~ is executed , said applications within a container ~~have~~ has no access to system files or applications in other containers or to system files within the operating system during execution thereof ~~if those applications or system files are read/write files.~~

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

With regard to the patentability of Claim 10, this claim imports the patentable features of Claims 1 and 2 from which it depends. Furthermore, Claim 10 defines application software within a container being isolated from other containers. Forbes et al. do not have plural containers and therefore have no isolation between containers.

Claim 13 has been amended as shown and now defines:

A method as defined in claim 1 further comprising the step of associating with a plurality of containers a stored history of when processes related to applications within the container are executed for at least one of, tracking statistics, resource allocation, and for monitoring the status of the application.

Forbes et al. do not disclose a history of file usage or tracking of processes within an application. Once Forbes et al.'s application is executing, Forbes et al. make no suggestion or hint that there is interaction with the application. Interaction with the application is required to store container history of tracking statistics, or resource allocation, for monitoring the status of an application.

Claim 14 has been rejected as being anticipated by Forbes et al.

The Examiner has said that:

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

The teachings of Forbes et al (Forbes et al Abstract, col. 2, lines 33-col. 3, line 56, figures 1-4 and associated descriptions, whereas the use of a software package manager in a network environment using a distribution unit containing components for a software package and manifest file that supports the installation (i.e., 'creating containers prior to ... containers are created by ... determining which files are being used'), execution/runtime aspects of predetermined applications /associated services across all types of code/operating systems, so as to allow easily extending new code types as they arise (e.g., Forbes et al col. 2, lines 38-col. 3, lines 49), clearly encompasses the claimed limitations as broadly interpreted by the examiner.) suggest such limitations.

Amended Claim 14 defines: A method as defined in claim 1 comprising the step of creating containers prior to said step of storing containers in memory, wherein containers are created by:

- a) running an instance of a service on a server;
- b) determining which files are being used; and,
- c) copying applications and associated system files to memory without overwriting the associated system files so as to provide a second instance of the applications and associated system files.

Claim 14 with the claim limitations of amended Claim 1 from which it depends defines:

14. (amended) In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel **a set of associated local system files compatible** with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications may be executed in a secure environment, wherein the applications each include an object

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

executable by at least some of the different operating systems for performing a task related to the service, the method comprising the steps of:

storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers; wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel, ~~and~~ wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server prior to said storing step., wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server, and wherein the application software cannot be shared between the plurality of secure containers of application software, and wherein each of the containers each has a unique root file system that is different from the operating system's root file system; and,

comprising the step of creating containers prior to said step of storing containers in memory, wherein containers are created by:

- a) running an instance of a service on a server;
- b) determining which files are being used; and,
- c) copying applications and associated system files to memory without overwriting the associated system files so as to

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

provide a second instance of the applications and associated system files.

The Examiner suggests in the paragraph below, taken from the Office Action, that Claim 14 is anticipated because:

The teachings of Forbes et al (Forbes et al Abstract, col. 2, lines 33-col. 3, line 56, figures 1-4 and associated descriptions, whereas the use of a software package manager in a network environment using a distribution unit containing components for a software package and manifest file that supports the installation, execution/runtime aspects of predetermined applications /associated services across all types of code/operating systems, so as to allow easily extending new code types (i.e., 'assigning an identity to the containers ... containers are created by ... including at least one of ... a unique IP address ... installing the container ... testing the applications and files ') as they arise (e.g., Forbes et al col. 2, lines 38-col. 3, lines 49), clearly encompasses the claimed limitations as broadly interpreted by the examiner.) suggest such limitations.

It should become readily apparent that the reference to citations taken from Forbes et al. in the paragraph above do not disclose the method steps within amended Claim 14. If the Examiner disagrees, Applicants would be appreciative if the Examiner would indicate specifically by indicating the line numbers and specific passage(s) where each claimed feature exists within the Forbes et al., patent.

Claim 15 defines: A method as defined in claim 14 comprising the steps of:
assigning an identity to the containers including at least one of a unique IP address, a unique Mac address and an estimated resource allocation;
installing the container on a server; and,

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

testing the applications and files within the container.

Claim 15 imports the limitations of Claim 14, defined above, and further defines steps of assigning an identity to the containers. It should be noted that Forbes et al. do not have plural containers. Furthermore, assigning a unique IP address or Mac Address to plural containers is not suggested in Forbes et al. as they merely run a single instance of a software application. They do not have plural containers each having a software application each with an associated IP or Mac Address associated with a respective container where the software resides.

Forbes et al. teach that a software distribution mechanism is used with a networked system environment. The application installed, by means of Forbes et al., takes on the identity, defined as one or more of IP address, MAC, hostname, of the underlying OS. Therefore, the application installed by Forbes et al. assumes the identity supplied by the OS. The instant invention claims that an application associated with a container assumes the identity, of the IP address, or MAC, hostname, from container with which it is associated and resides in, and not from the OS.

After a careful reading of the Forbes et al. patent, it becomes evident that there is no mention of an application having an associated network identity. Forbes et al. teach that a manifest file can reside on network store; files can be accessed from a network. However, there is no teaching that an application assumes a network identity.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

Claim 16 defines: A method as defined in claim 1 comprising the step of creating containers prior to said step of storing containers in memory, wherein a step of creating containers includes:
using a skeleton set of system files as a container starting point and installing applications into that set of files.

This claim is clearly novel as Forbes et al. do not create or provide plural containers as defined in Claim 1 or 16. In view of this, the specific implementation defined in Claim 16 is not suggested by Forbes et al.

Claim 17 has been rejected under 35 U.S.C. 102 (b) for the same reasons as Claim 1 has been rejected. Claim 17 has been amended in a similar fashion to the amendments of Claim 1.

Claim 17 defines a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers, each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers, each container of files is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac_address; wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes wherein the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system, each container having its own execution file

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

associated therewith for starting one or more applications, in operation, each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and, a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.

Applicants have taken the liberty of re-writing Claim 17, more clearly illustrating the features within the claim.

- A system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers,
- each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers,
- each container of files is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac_address;
- wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes
- wherein the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system,

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

- each container having its own execution file associated therewith for starting one or more applications, in operation,
 - each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and,
 - a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.
- Forbes et al. do not have a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers; in fact Forbes et al. make no mention of containers at all, and if one were to incorrectly construe Forbes et al. as having a secure container for containing their software, this would in no way be a teaching of a plurality of secure containers having the features which follow in Claim 17.
- Forbes et al. do not teach that each container is mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers. Since Forbes et al. make no mention of plural containers, they clearly do not teach mutual exclusivity between containers and make no reference to an absence of file sharing of read/write files with other containers.
- Forbes et al. do not teach containers so they clearly do not teach or suggest that each container of files is said to have its own unique identity associated therewith,

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

said identity comprising at least one of an IP address, a host name, and a Mac address.

- Forbes et al. do not teach that the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes, as Forbes et al. do not teach a plurality of containers.
- Forbes et al. replace and overwrite their software with a latest version so they teach away from the limitation that the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system.
- Forbes et al. do not teach plural containers wherein each container having its own execution file associated therewith for starting one or more applications, in operation. Forbes et al. only teach a single instance of a single software application.
- Forbes et al. do not teach plural containers wherein each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and,
- a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

After thoroughly reading the Forbes et al. cited patent and considering the invention defined in Claim 17, it becomes evident that Forbes et al. do not disclose each and every element in Claim 17. In fact, it is questionable whether Forbes et al. disclose any of the patentable elements within Claim 17. In summary, Forbes et al. do not define multiple containers and Forbes et al. do not define any isolation of files. Forbes et al. teach that more than one software distribution unit exists and that files may be shared. In Forbes et al.'s teaching, a distribution replaces an older version and is not isolated.

Claim 18 of the instant invention defines: A computing system as defined in claim 17, further comprising a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources.

Forbes et al. do not have a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources.

In the Forbes teaching, a single application starts and there is no further interaction with the application by the package manager taught by Forbes et al. According to the invention defined by Claim 18 there is a scheduler in addition to the operating system that interacts with the application while it is executing.

Claim 19 has been rejected under 35 U.S.C. 102 (b) as being anticipated by Forbes et al.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

The Examiner has stated:

The teachings of Forbes et al (Forbes et al Abstract, col. 2, lines 33-col. 3, line 56, figures 1-4 and associated descriptions, whereas the use of a software package manager in a network environment using a distribution unit containing components for a software package and manifest file that supports the installation, execution/runtime aspects of predetermined applications /associated services across all types of code/operating systems, so as to allow easily extending new code types as they arise (e.g., Forbes et al col. 2, lines 38-col. 3, lines 49), clearly encompasses the claimed limitations as broadly interpreted by the examiner.) suggest such limitations.

Applicants have carefully reviewed the above mentioned Abstract, Col. 2 to Col. 3 and figures 1-4 and respectfully disagree with the Examiner as to a finding of anticipation. For a claim to be anticipated each and every element must be present.

Claim 19, including the limitations of amended Claim 17 defines:

- A system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers,
- each container being mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers,
- each container of files is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac_address;

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

- wherein, the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes
- wherein the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system,
- each container having its own execution file associated therewith for starting one or more applications, in operation,
- each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and,
- a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications,
- wherein the run time module includes an intercepting module associated with the plurality of containers for intercepting system calls from any of the plurality of containers and for providing values alternate to values the kernel would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of the IP address, the host name and the Mac_Address; and
- further comprising a scheduler comprising values related to an allotted time in which processes within a container may utilize predetermined resources.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

Forbes et al. do not teach:

- Forbes et al. do not have a system having a plurality of secure containers of associated files accessible to, and for execution on, one or more servers; in fact Forbes makes no mention of containers at all, and if one were to incorrectly construe Forbes as having a secure container for containing his software, this would in no way be a teaching of a plurality of secure containers having the features which follow in claim 17.
- Forbes et al. do not teach that each container is mutually exclusive of the other, such that read/write files within a container cannot be shared with other containers. Since Forbes et al. make no mention of plural containers they clearly do not teach mutual exclusivity between containers and make no reference to an absence of file sharing of read/write files with other containers.
- Forbes et al. do not teach containers so they clearly do not teach or suggest that each container of files is said to have its own unique identity associated therewith, said identity comprising at least one of an IP address, a host name, and a Mac address.
- Forbes et al. do not teach that the plurality of files within each of the plurality of containers comprise one or more application programs including one or more processes, and associated system files for use in executing the one or more processes, as Forbes et al. do not teach a plurality of containers.

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

- Forbes et al. replace and overwrite their software with a latest version so they teach away from the limitation that the associated system files are files that are copies of files or modified copies of files that remain as part of the operating system.
- Forbes et al. do not teach plural containers wherein each container having its own execution file associated therewith for starting one or more applications, in operation. Forbes et al. only teach a single instance of a single software application.
- Forbes et al. do not teach plural containers wherein each container utilizing a kernel resident on the server and wherein each container exclusively uses a kernel in an underlying operation system in which it is running and is absent its own kernel; and,
- a run time module for monitoring system calls from applications associated with one or more containers and for providing control of the one or more applications.
- AND more particularly Forbes et al. do not teach a run time module including an intercepting module associated with the plurality of containers for intercepting system calls from any of the plurality of containers and for providing values alternate to values the kernel would have assigned in response to the system calls, so that the containers can run independently of one another without contention, in a secure manner, the values corresponding to at least one of the IP address, the host name and the Mac Address;

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

Since Forbes et al. do not teach a system wherein there is inception of system calls and subsequent interaction with the application after it is launched they do not teach system calls. More importantly, Forbes et al. do not include any hint or suggestion of having interception of system calls.

Having system calls advantageously provides the ability to monitor the application as it executes and change parameters that are provided to and received from the operating system, thereby altering interaction with the operating system so as to allow the application to execute in an isolated and contained environment.

Claim 20 is said to be anticipated by Forbes et al. for essentially the same reasons as Claim 17.

As Applicants have pointed out above, Claim 17 is patentable and therefore Claim 20 at least imports the patentable features of Claim 17. Furthermore, Claim 17 defines that the run time module performs:

- monitoring resource usage of applications executing;
- intercepting system calls to kernel mode, made by the at least one respective application within a container, from user mode to kernel mode;
- comparing the monitored resource usage of the at least one respective application with the resource limits; and,
- forwarding the system calls to a kernel on the basis of the comparison between the monitored resource usage and the resource limits.

Interception of system calls is not described anywhere within the Forbes et al. patent. Forbes et al. do not monitor

In re Patent Application of:

ROCHETTE ET AL.

Serial No. **10/939,903**

Filed: **09/13/2004**

after their software execution begins. Therefore they do not monitor/intercept/compare/ and forward system calls. When the Forbes et al. software is fully loaded, it executes. In the instant invention, after execution begins, the steps of monitoring/intercepting/comparing and forwarding system calls is performed. No such steps are taught by Forbes et al.

In view of the foregoing amendments to the claims and remarks, it is respectfully submitted that the instant application is now in condition for allowance.

Early and favorable reconsideration of the application would be appreciated.

Should any minor informalities need to be addressed, the Examiner is encouraged to contact the undersigned attorney at the telephone number listed below.

Please charge any shortage in fees due in connection with the filing of this paper, including Extension of Time fees, to Deposit Account No. 50-1465 and please credit any excess fees to such deposit account.

Respectfully submitted,

/CHRISTOPHER F. REGAN/

Christopher F. Regan

REG. NO. 34,906

Customer No.: 27975

Telephone: (407) 841-2330

Date: September 3, 2008